Saving Bandwith and Supporting Cryptography in the Internet : An Application of Wavelet Filters

João C. DOVICCHI
Center for Sonic Computing and Multimedia, Uberlândia Federal University
Uberlândia - MG - Brazil

and

Jamil S. BARBAR
Faculty for Computing, Uberlândia Federal University
Uberlândia - MG - Brazil

Abstract

Compression and cryptography of digital data for network transmission using digital audio signals and its wavelet tranforms is described in this paper. Wavelts are mathematical functions which can amplify data intervals, splitting them into different segments off different frequencies. This treatment allows us to perform data analysis and compression on each corresponding scale. A Wavelet transform in a space of some specific data function can sparsely represent each subspace of this function in the "time-frequency" domain. This property allows us to use it for data compression, and once this compression is biorthogonally dependent of the filter coefficients it can be used as a cryptographic tool where those coefficients can represent pairs of public and prived keys. In this work we describe a new method to compress and encrypt digital data, simultaneously, for transmission over a network environment. Although results were yelded for digital audio data the method can easily be portable to other digital data issues.

Keywords: Wavelets, compression, cryptography, transmission bandwidth, Internet security.

1 Introduction

Internet security and bandwidth savings are important issues on today's data communications. The rapidly growing use of the Internet as a communication tool doubles its traffic every half year. This means that in less than one year we will need more than 35 Tpbs in communication capacity. Naturally, data trafic will grow more quickly than voice in future communications and new transport infrastructure should be otimized for data [5]. But, besides any network infraestructure impprovement, bandwidth control and security issues will demmand researches for data encryption and compression. In real time applications, these algorithms must reflect a quick and efficient method.

Wavelet transforms can be applied to audio data compression because it occurs natturally during signal thresholding which can be combined to a *Huffman entropy coding* with quite impressive results [1, 2, 11]. Nevertheless it works very well for digital audio and image files, its application to other digital data types, namely those with poor frequency dependence, is not trivial. But, since entropy coding can be used to code information efficiently [12], a combined lookup table (coding table) can be constructed under a wavelet base thresholding data and converting it into a sparse vector suitable for any compressing algorithm. Thus, Wavelets coefficients can be used to crypt and compress digital data at the same time.

The discrete wavelet transform can represent a signal $x(k)$ as a diadic and orthogonal projection $X_{n,m}(k)$ over a set of bandpass sequences $\psi_{n,m}(k) = \psi_{n,0}(k-2^n m)$, where $n = 1, 2, \ldots$ represents the scale and $m = 0, 1, 2, \ldots$ represents time shifts. We could then notate:

$$x(k) = \sum_{n=1}^{N} \sum_{m} X_{n,m} \psi_{n.m}(k) \qquad (1)$$

where,

$$X_{n,m} = \sum_k x(k)\psi_{n,m}(k) \qquad (2)$$

is the wavelet transform of $x(k)$.

A pair of transfer functions $\{H(z), G(z)\}$, where $H(z)$ is the low-pass and $G(z)$ is the high-pass, can be obtained from a set of coefficients $h(k)$that forms an orthonormal basis for the transformation and have some properties:

$$\sum_{-\infty}^{+\infty} h_k = \sqrt{2} \qquad , \qquad (3)$$

and

$$\sum_{-\infty}^{+\infty} |h_k|^2 = 1 \qquad . \qquad (4)$$

1.1 Parametric values as encryption factors

Wavelets coefficients can be determined by a group of parametric equations. Daubechies [4] has described Pollen's parametric equations [10] that can be used in 2, 4 or 6 wavelet coefficients (see also Cody's paper [3]):

$$\begin{aligned}
h_{-2} &= \frac{(1+\cos(\alpha)+\sin(\alpha))\cdot(1-\cos(\beta)-\sin(\beta))+2\sin(\beta)\cos(\alpha)}{4} \\
h_{-1} &= \frac{(1-\cos(\alpha)+\sin(\alpha))\cdot(1+\cos(\beta)-\sin(\beta))-2\sin(\beta)\cos(\alpha)}{4} \\
h_0 &= \frac{1+\cos(\alpha-\beta)+\sin(\alpha-\beta)}{2} \\
h_1 &= \frac{1+\cos(\alpha-\beta)-\sin(\alpha-\beta)}{2} \\
h_2 &= 1 - h_{-2} - h_0 \\
h_3 &= 1 - h_{-1} - h_1
\end{aligned}$$

TeKolste [15], pointed the way to generate all compactly-supported real valued wavelets. The algorithm for finding the dilation coeeficients from his his parameters is described as follows:

If $\{a_i | i \in Z\}$ is a series, he calls a "pivot" of α at k as $\{b_i | i \in Z\}$ where b_i is determined by

$$b_i = a_i \cos(\alpha) + (-1)^i a_{2*k+1-i} * \sin(\alpha) \qquad (5)$$

A wavelet of dimension d, which has zero coefficients outside the interval $[0, \ldots, 2d+1]$ are parametrized by d angles $[\alpha_1, \alpha_2, \ldots, \alpha_d]$ and can be coded as:

```
Let alpha_0 = pi/2 - (alpha_1 + ... + alpha_d)
Let a_0,0 = cos(alpha_0/2), a_0,1 = sin(alpha_0/2)
For j = 0, ..., d-1
   let a_j+1 = {a_(j+1,i) | i in Z}
```

$a_{j+1} = \{a_{j+1,i} | i \in Z\}$ is the series obtained from $a_j = \{a_(j,i) | i \in Z\}$ by pivoting through α_{j+1} at $j+1$. Then a_d is a valid set of dilation coefficients.

Once we have the possibility to construct any set of valid wavelet coefficients with compact support, these coefficients can be used as encryption factors of the signal. Since it has compact support we can expect that the coded (wavelet transformed) signal can be more efficiently compressed.

1.2 Lifting and efficiency

In the last decade, numerical methods and mathematical theories of wavelet transforms opened new paths for analysis, compression and recovery of digital data. Nevertheless, only a few years ago Swelden's lifting method [13, 14] improved computational calculus using in loco variables which implies in low memory requirements and high computing efficiency.

The use of *lifting schema* during wavelet transform can still optimize efficiency, and memory and time savings impproving real time application. Note that compress data transmission can also compensate algorithm speed.

The idea behind the *lifting schema* is to compute the *in loco* transform, which means no auxiliary memory allocation. Combining the wavelet transform and lifting algorithm we can easily design transfer functions, since everything occurs in the time (or space) domain. Thus, the signal reconstruction is frequency independent and has the following advantages:

1. *In loco* calculation: all calculations can be done on the same memory position with no auxiliary memory necessary;

2. Efficiency: reduced operations with `float` variables once we can reuse subexpressions;

3. Paralelism: Transform schem shows an inherent paralelism capability in all scales;

2 Encryption Algorithm

Matsui's algorithm [7] can be used to generate a cryptographic key which can be used to generate a set of coefficient of the dilation equation for the Wavelet Transform. This algorithm is divided into two parts: first "key scheduling" and second "data randomizing".

Key scheduling uses a 128-bit input key to produce a 128-bit expanded key. Data randomizing part takes a 64-bit input data and mixes it, producing the encryption. Processing data randomizing part in reverse order yields the input data, causing data decryption.

2.1 Key Scheduling

Key scheduling part consists of the following operations.

```
for i = 0, ..., 7 do
   EK[i] = K[i*2]*256 + K[i*2+1];
for i = 0, ..., 7 do
   EK[i+ 8] = FI(EK[i], EK[(i+1)%8]);
   EK[i+16] = EK[i+8] & 0x1ff;
   EK[i+24] = EK[i+8] >> 9;
```

Where K is an input key, and each element of $K[i]$, contains an 8-bit segment of the key, respectively. EK denotes an expanded key, and each element $EK[i]$, contains a 16-bit segment of the expanded key. Input data of $K[0], \ldots, K[15]$ are copied to $EK[0], \ldots, EK[7]$. Expanded key is produced from $EK[0], \ldots, EK[7]$ by using function FI, and stored in $EK[8], \ldots, EK[15]$. Function FI is described in the following section.

2.2 Data Randomizing

Data randomizing part uses two kinds of function, which are called function FO and function FL. Function FO calls another function, namely FI. The key expansion part also uses function FI. Function FI uses two S-boxes, namely $S7$ and $S9$. Each function is described as follows.

Function FO takes two parameters. One is a 32-bit width input data, FO_{IN}. The other is an index of EK, namely k. And FO returns a 32-bit width data, namely FO_{OUT}.

```
FO(FO_IN, k)
   var t0, t1 as 16-bit integer;
      t0 = FO_IN >> 16;
      t1 = FO_IN & 0xffff;
      t0 = t0 ^ EK[k];
      t0 = FI(t0, EK[(k+5)%8+8]);
      t0 = t0 ^ t1;
      t1 = t1 ^ EK[(k+2)%8];
      t1 = FI(t1, EK[(k+1)%8+8]);
      t1 = t1 ^ t0;
      t0 = t0 ^ EK[(k+7)%8];
      t0 = FI(t0, EK[(k+3)%8+8]);
      t0 = t0 ^ t1;
      t1 = t1 ^ EK[(k+4)%8];
      FO_OUT = (t1<<16) | t0;
      FO_OUT = (t1<<16) | t0;
```

Function FI takes two parameters. One is a 16-bit width input data, namely FI_{IN}. The other is a part of EK, namely FI_{KEY}, which is also 16-bit width. And FI returns a 16-bit width data, namely FI_{OUT}.

```
FI(FI_IN, FI_KEY)
    var d9 as 9-bit integer;
    var d7 as 7-bit integer;
       d9 = FI_IN >> 7;
       d7 = FI_IN & 0x7f;
       d9 = S9TABLE[d9] ^ d7;
       d7 = S7TABLE[d7] ^ d9;
       ( d7 = d7 & 0x7f; )
       d7 = d7 ^ (FI_KEY >> 9);
       d9 = d9 ^ (FI_KEY & 0x1ff);
       d9 = S9TABLE[d9] ^ d7;
       FI_OUT = (d7<<9) | d9;
    return FI_OUT;
```

S7-Table and S9-Table denote the S-boxes $S7$ and $S9$ respectively in terms of look up table notation. For tables description he description see Matsui's paper [7].

Function FL takes two parameters. One is a 32-bit data, namely FL_{IN}. The other is an index of EK, namely k. And FL returns a 32-bit width data, namely FL_{OUT}.

```
FL(FL_IN, k)
var d0, d1 as 16-bit integer;
d0 = FL_IN >> 16;
d1 = FL_IN & 0xffff;
if (k is an even number) then
   d1 = d1 ^ (d0 & EK[k/2]);
   d0 = d0 ^ (d1 | EK[(k/2+6)%8+8]);
else
   d1 = d1 ^ (d0 & EK[((k-1)/2+2)%8+8]);
   d0 = d0 ^ (d1 | EK[((k-1)/2+4)%8]);
endif
FL_OUT = (d0<<16)  d1;—
return FL_OUT;
```

When the algorithm is used for decryption, function $FLINV$ is used instead of function FL.

```
FLINV(FL_IN, k)
var d0, d1 as 16-bit integer;
d0 = FL_IN >> 16;
d1 = FL_IN & 0xffff;
if (k is an even number) then
   d0 = d0 ^ (d1 | EK[(k/2+6)%8+8]);
   d1 = d1 ^ (d0 & EK[k/2]);
else
   d0 = d0 ^ (d1 | EK[((k-1)/2+4)%8]);
   d1 = d1 ^ (d0 & EK[((k-1)/2+2)%8+8]);
endif
FL_OUT = (d0<<16) | d1;
return FL_OUT;
```

In most cases, data randomizing part consists of 8 "steps". Each step contains the call of function FO. Additionally, even-number round includes the calls of function FL. After the final round, all FL functions are called again. The detail description is as follows.

64-bit plaintext P is divided into the leftmost 32-bit $D0$ and the rightmost 32-bit $D1$.

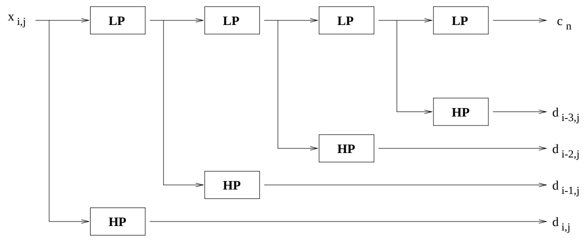

Figure 1: The piramid algorithm of the forward Fast Wavelet Transform (FWT) and the repeated convolution of the sampled data with the wavelet coefficients Low-Pass (LP) and High-Pass (HP) filters.

```
D0 = FL(D0, 0); //step 1
D1 = FL(D1, 1);
D1 = D1 ^ FO(D0, 0);
D0 = D0 ^ FO(D1, 1); //step 2
D0 = FL(D0, 2); //step 3
D1 = FL(D1, 3);
D1 = D1 ^ FO(D0, 2);
D0 = D0 ^ FO(D1, 3); //step 4
D0 = FL(D0, 4); //step 5
D1 = FL(D1, 5);
D1 = D1 ^ FO(D0, 4);
D0 = D0 ^ FO(D1, 5); //step 6
D0 = FL(D0, 6); //step 7
D1 = FL(D1, 7);
D1 = D1 ^ FO(D0, 6);
D0 = D0 ^ FO(D1, 7); //step 8
D0 = FL(D0, 8); // final step
D1 = FL(D1, 9);
```

64-bit ciphertext C is constructed from $D0$ and $D1$ as following operation.

```
C = (D1<<32) | D0;
```

3 Methods

Using Mitsui's algorithms we generate a 128 bit key which was used to build 16 integer dilation coefficients for the Wavelet filter as described by TeKolste [15]. Samples (x_{ij}) where first converted into a 2^n bytes with addition zeroes and then treated with this filter up to 4 iterations yielding 4 high-pass ($d_{i-n,j}$)and 1 low-pass (c_n) blocks (cf fig. 1).

The first iteration results in two 2^{n-1} blocks and each iteration decimates the remaining low-pass by a factor of 2. For analysis purposes, each segment was separated and studied individually in order to find the compression factor for each of them. Each one were compressed using a Huffmann entropy table as a lookup table as described by Bradley [1].

4 Results and conclusions

We noticed that the first step high-pass segment presented, as we have expected, the greater compression factor, since it is the most sparse segment (cf. table 1). As we continue the iteration process, each new high pass segment presented a decreasing compression factor, for they are less and less sparse.

Segment data	**Relative size**	**Mean % compression**
$d_{i,j}$	2^{n-1}	92
$d_{i-1,j}$	2^{n-2}	68
$d_{i-2,j}$	2^{n-3}	47
$d_{i-3,j}$	2^{n-4}	22
c_n	2^{n-4}	4

Table 1: Mean percent of compression of each segment after 4 iteration Wavelet transform of crypted audio files

Using a proportional factor for the compression based on each block size we can get the mean value of the total compression.

$$M = \frac{2^{n-4}f(c_n) + \sum_{k=1}^{4} 2^{n-k} f(d_{i-k-1,j})}{T} \qquad (6)$$

where M is the mean value of compression, $f(\cdot)$ is the factor of compression for the segment and T is the total size of the sample. Using equation (6) we obtain a mean of 70.5 % of total compression.

4.1 Time saving considerations

Since the algorithm for crypting and compressing occurs at the same time, and the Wavelet transform using the lifting schema is very fast, we believe that any time loss during the crypt/compression process can be overtaken during compressed data transmission.

Although we have tried to test with other non sound binary data, we are still working on the selection of ideal dilation equation coefficients to accomplish this task.

4.2 Security considerations

The algorithm, which is described in this document, is designed in terms of the theory of provable security against differential cryptanalysis and linear cryptanalysis [6, 8, 9]. According to recent results, when the algorithm consists of 8 rounds, both differential characteristic probability and liner characteristic probability are 2^{-140}. For reference, probabilities of DES are 2^{-62} and 2^{-46}, respectively.

5 Acknowledgements

The authors wish to thank the staff and students of the Sonic Computing Lab and Prof. Waldenor Moraes Filho for the kind reading and correction on the English Language. This work was supported by Brazilian Federal Government and Uberlândia Federal University Foundation.

References

[1] Bradley, J. N.; Brislawn, C. M. & Hopper, T. 1993. "The FBI wavelet / scalar quantization standard for grey-scale fingerprint image compression." *Visual Information Processing II* : *Proc. SPIE*, Orlando, Florida, pp. 293-304.

[2] Brislawn, C. M. 1995. "Fingerprints go digital", *Notices of the A. M. S.*, 42 (11), pp. 1278-1283.

[3] CODY, M. A. 1992. "The Wavelet Transform Beyond Fourier Transforms." *Dr. Dobb's Journal*, M & T Publishing Inc., April, pp. 12-24.

[4] Daubechies, I. 1992. **Ten Lectures on Wavelets**, Soc. Ind. Appl. Math. (SIAM), ISBN 0-89871-274-2.

[5] Dixit, S. & Ye, Y. 2001."Streamlining the Internet-Fiber Connection", *IEEE Spectrum*, 38 (4), (April), pp.52-57.

[6] Matsui, M. 1996. "New Structure of Block Ciphers with Provable Security Against Differential and Linear Cryptanalysis", Fast Software Encryption - Third International Workshop, LNCS 1039, Springer Verlag, pp.205-218.

[7] Matsui, M. 1997 "New Block Encryption Algorithm MISTY", Fast Software Encryption - 4th International Workshop (FSE'97), LNCS 1267, Springer Verlag, pp.54-68.

[8] Nyberg, K. & Knudsen, L. R. 1995 "Provable Security Against a Differential Attack", Journal of Cryptology, Vol.8, No.1, pp. 27-37.

[9] Nyberg, K. 1995 "Linear Approximation of Block Ciphers", Advances in Cryptology - Eurocrypt'94, LNCS 950, Springer Verlag, pp.439-444.

[10] Pollen, D. 1989. "Parametrization of Compactly Supported Wavelets." *Tech. Rep.*, Aware Inc.

[11] Schröder, P. 1995. "Wavelet image compression: beating the bandwidth bottleneck.", *Wired*, (May), p. 78.

[12] Stinson, D. R. 1995. **Cryptography: Theory and Practice**, Boca Raton: CRC Press.

[13] Sweldens, W. 1995. "The Lifting Scheme: A Construction of Second Generation Wavelets.", *Tech. report 1995:6*, Industrial Mathematcs Iniciative, Department of Mathematics, University of South Carolina.

[14] Sweldens, W. 1996. "The Lifting Scheme: A Custom-design Construction of Biorthogonal Wavelets.", *Appl. Comput. Harmon. Anal.*, 3(2):186-200.

[15] TeKolste, K. 2000. "Personal communication: Parametrization of Biorthogonal Wavelets", *Wavelets Digest Discussion Group* WD 4.10 #24.
`http://www.wavelet.org`